Leader Election in a Ring

Reference: Attiya and Welch, Sections 3.1-3.3.

Problem Elect a leader. Every process outputs 0 or 1, exactly 1 process outputs 1.

If processes are identical (i.e., if they don't have unique ids, and they all execute identical programmes), Leader election (LE) is impossible in a ring (using a deterministic algorithm), even if the ring is synchronous. [Angluin 1980]

At each time step, every process will be in identical state, so they will never be able to break symmetry. All processes produce the same output. (A formal proof could be done by induction: Suppose all nodes are in the same state at the start of some step. Every node receives identical messages during that step (because every node's neighbours were in the same state at the start of the step, so if one sends a message, they all do). So every node undergoes the same state transition during the step and every node ends up in the same state at the end of the step.

What if network is asychronous ? Same argument applies to show impossibility, because synchronous ring is more powerful than asynchronous.

Problem is solvable if processes all have unique ids, even for asynchronous model. Choose process with maximal id as leader.

Idea 1. Each process will send a message (containing its own id) all the way around the ring to see if there is a bigger id.

Total number of messages is $O(n^2)$ (since each of n processes sends out a message that has to traverse all the n nodes of the ring).

We would like to decrease the message complexity. Some processes are clearly not going to win because they have really small ids, so we would like to eliminate them early without requiring them to send a message all the way around the ring.

Better Idea. First elect local leaders who will compete with one another for leadership of larger and larger sections of the ring.

In phase i, elect leaders in sections of length 2^i. Send out message to distance of 2^{i-1} on either side with your id (using hop-counts to control the distance). When messages come back they'll tell you whether you have largest id in that section of the ring. Only those processes who won in phase i go on to participate in phase $i+1$. When your own message goes all the way around the ring (i.e. $2^{i-1} \geq n$) then you know you can stop and produce your output. Any process who loses in any phase can output 0.

How many messages does it take ?

In phase i the number of messages per participating process is 2^{i+1}. Between any two participating processes there must be at least 2^{i-2} non-participating processes, since each participating process has a bigger id than any process within distance 2^{i-2} of it. Number of participating processes is $\leq \frac{n}{2^{i-2}+1} \leq \frac{n}{2^{i-2}}$. So the number of messages in phase i is $\leq 8n$. There are $\log n$ phases. Total message complexity $\in O(n \log n)$.

These 2 results show a separation between models -

1. identical processes in synchronous ring;
2. non-identical processes in synchronous ring.

Solvable in (1) $\Rightarrow$ solvable in (2) (just ignore the id-s). Solvable in (2) $\not\Rightarrow$ solvable in (1) (LE is an example). Model (2) is strictly stronger than (1).

Can we improve algorithm complexity ? No. We'll show that any LE algorithm in a ring needs $\Omega(n \log n)$ messages in worst case.

Variant of LE . LE$'$ - every process must output maximal process id in the system. We'll prove lower bound for this version.

(The lower bound for the standard version of LE follows: see Homework Exercise #2).

Assumptions -

- processes don't know size of the ring;
- asynchronous ring. (Note: if we prove lower bound for asynchronous ring, the lower bound must also hold for synchronous rings, since the syncrhonous model is more powerful.)

Show how to construct an execution that uses $\Omega(n \log n)$ messages. We'll do it inductively by glueing together costly executions to get even more costly ones. The following claim proves the bound if n is a power of 2, and generalizing to other values of n is not too difficult.

Claim Consider any algorithm that solves LE'. If n is a power of 2 $\exists$ execution of the algorithm on any ring of n processes where no message goes across some edge and at least $M(n)$ messages are sent, where
$M(2) = 1$ and
$M(n) = 2M(\frac{n}{2}) + \frac{n}{4} - 1$.

(Note: $M(n) \in \Theta(n \log n)$, by the Master Theorem).

Proof.

$n = 2$ is a trivial case. LE' clearly requires at least one message in a ring of 2 processes.

Inductive step.

Suppose claim is true for $n = 2^k$. Prove claim holds true for $n = 2^{k+1}$. Consider 2 rings, R_1 and R_2 of size 2^k. (R_1,R_2 have disjoint sets of id-s.) (See figure 1). By inductive hypothesis, $\exists$ execution on R_i where some edge e_i is not used and number of messages in execution α_i is $\geq M(2^k)$.

Consider ring R of size 2^{k+1} shown in figure 2. It consists of a copy of R_1 and R_2 with e_1 and e_2 removed, and two new edges e and e' added to connect them as shown. Construct bad execution in new ring. First, run α_1 on R_1 portion, and α_2 on R_2. Notice that processes in R_i cannot tell during α_i whether they are executing in R_i or in R, because no message travels across e_i, e or e'. So far, we have an execution with at least $2M(2^k)$ messages. Continue running algorithm

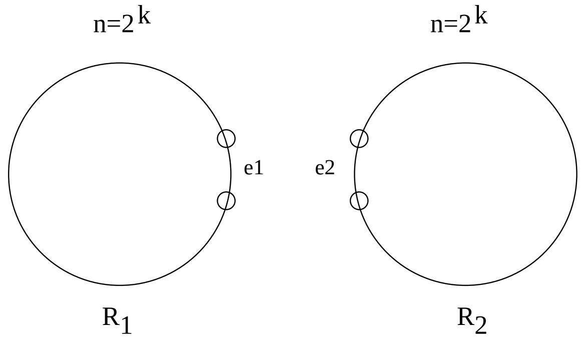

Figure 1: Two separate rings

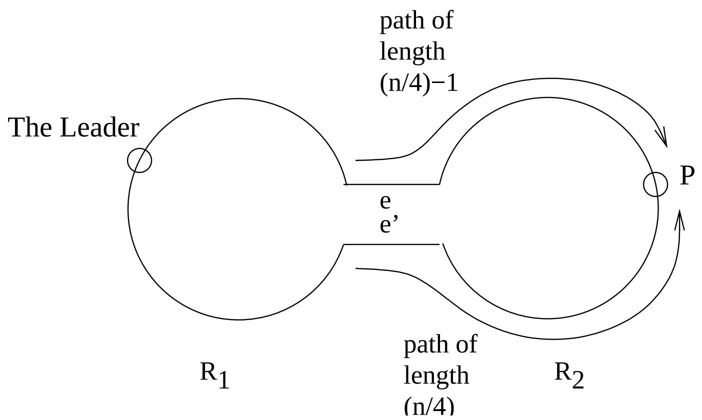

Figure 2: R1 and R2 together form a bigger ring

on R (without allowing any messages across e and e') until messages stop happening. Goal - get execution with $\geq 2M(2^k) + \frac{2^{k+1}}{4} - 1$ messages. (If this continuation generates $n/4 - 1$ messages, we're done. Otherwise we'll continue as follows.)
Let α be the execution on R so far.

Imagine continuing the execution, starting with α until completion. Without loss of generality let's assume the leader is in R_1. Process P, which is diametrically opposite from e and e' in R_2, has to receive info from R_1 before it can terminate. So some chain of messages (of length $\geq \frac{n}{4} - 1$) must have reached P either from e or from e'. (See figure 2). We create the costly execution by starting with α and then running the algorithm without allowing any messages across the *other* edge. Then $\geq \frac{n}{4} - 1$ messages are still generated after α, and the total number of message inthis execution is at least $2M(n/2) + n/4 - 1$.